to appear in ApJS v145, March 2003

Tree–Particle–Mesh: an adaptive, efficient, and parallel code for collisionless cosmological simulation

Paul Bode and Jeremiah P. Ostriker

Princeton University Observatory, Princeton, NJ 08544-1001

ABSTRACT

An improved implementation of an N-body code for simulating collisionless cosmological dynamics is presented. TPM (Tree–Particle–Mesh) combines the PM method on large scales with a tree code to handle particle-particle interactions at small separations. After the global PM forces are calculated, spatially distinct regions above a given density contrast are located; the tree code calculates the gravitational interactions inside these denser objects at higher spatial and temporal resolution. The new implementation includes individual particle time steps within trees, an improved treatment of tidal forces on trees, new criteria for higher force resolution and choice of time step, and parallel treatment of large trees. TPM is compared to P^3M and a tree code (GADGET) and is found to give equivalent results in significantly less time. The implementation is highly portable (requiring a Fortran compiler and MPI) and efficient on parallel machines. The source code can be found at http://astro.princeton.edu/∼bode/TPM.

Subject headings: methods: N-body simulations — methods: numerical — cosmology: dark matter — cosmology:large-scale structure of universe

1. Introduction

Numerical simulation of the nonlinear evolution of collisionless dark matter, usually by following a set of point masses as they move under their mutual gravitational influence, has, over the last few decades, played an important role in increasing our understanding of cosmological structure formation. Rather than attempt to make a summary of the extensive literature available on this subject here, we direct the reader to the recent reviews by Bertschinger (1998) and Klypin (2000).

A search of the internet will reveal a diverse set of N-body codes now available to be downloaded by those interested in carrying out cosmological simulations. One of the first to be developed is the direct code of Aarseth (1999). The operation count for implementing any such particle–particle code scales as N^2, where N is the number of particles. One can reach larger N and thus higher mass resolution (but spatial resolution limited by the cell size on a grid) in the same amount of CPU time

using the PM (Particle–Mesh) method, for example the code by Klypin & Holtzman (1997). The operation count here scales as $N\log N$, with a small prefactor; PM is implemented on regular grids, where efficient fast Fourier transform (FFT) algorithms are available. To attain higher, subgrid resolution one can add to a PM code particle–particle interactions and refined subgrids, such as in the Hydra code (Couchman, Thomas & Pearce 1995). A gridless approach is used in tree codes, for example the ZENO package of Barnes (1998). The operation count in the Barnes–Hut tree algorithm (Barnes & Hut 1986) also scales as $N\log N$ but the prefactor is much larger— by a factor of 10–50 (Hernquist 1987)— than for a PM code. The tree code of Hernquist (1987) has been made parallel (Becciani & Antonuccio-Delogu 2001). A tree code with individual particle time steps called GADGET has recently been released in both serial and distributed memory parallel versions (Springel, Yoshida & White 2001). One promising new development is the use of multigrid methods involving mesh refinements of arbitrary shape (Kravtsov, Klypin & Khokhlov 1997), such as in MLAPM (Knebe, Green & Binney 2001). Another cell-based approach has recently been developed which shows improved performance over standard tree codes (Dehnen 2000, 2002) and is included as part of the NEMO package (Teuben 1995).

For a given choice of algorithm, a related problem is making it run efficiently on parallel computers (Yahagi, Mori & Yoshii 1999; Ricker, Dodelson & Lamb 2000; Viturro & Carpintero 2000; Lia & Carraro 2001; Springel, Yoshida & White 2001; Yahagi & Yoshii 2001; Dorband, Hemsendorf & Merritt 2002; Jing & Suto 2002; Miocchi & Capuzzo-Dolcetta 2002; Stadel 2002). Gravity is long-range, so to compute the force on a given particle, one needs some information from every other particle, and communication is thus required if particles are distributed among many processors. Xu (1995) presented a new algorithm, called Tree–Particle–Mesh, based on using domain decomposition in a manner designed to overcome this difficulty. TPM uses the efficient PM method for long-range forces and a tree code for sub-grid resolution. Isolated, overdense regions are each treated with a separate tree, thus ensuring coarse–grained parallelism.

Bode, Ostriker, & Xu (2000), hereafter BOX, made several improvements in the code of Xu (1995). In this paper we present further refinements which improve the accuracy and efficiency of the algorithm. These include the following: allowing individual particle time steps within trees, an improved treatment of tidal forces, new criteria for higher force resolution and for the choice of time step, and parallel treatment of large trees. In §2 we present an overview of latest version of TPM, including time stepping; §3 discusses implementation of domain decomposition and the particle time step criterion, with an overview of how the code works in practice. The accuracy of the code is explored in §4 by comparing results with two other algorithms: P^3M and the GADGET tree code. Concluding remarks and details of the public release of the TPM code are given in §5.

Calder et al. (2002) discuss the distinction between verifying a simulation (that is, being sure that the equations are being solved correctly) and validating it (having confidence that the equations and their solution actually resemble a real world problem). Obviously, in this paper only the former is done. Various aspects of the validity of cosmological simulations are addressed in Klypin (2000), Knebe et al. (2000), van Kampen (2000), Hamana, Yoshida & Suto (2002), Knebe (2002), Power

et al. (2002), Baertschiger, Joyce & Labini (2002), and references therein. When using an N-body code, the simulator will need to be cognizant of these issues.

2. Overview of TPM

TPM begins with a standard PM code (in fact, if there are no particles selected to be in trees, then TPM defaults to a PM code); this portion is similar to the PM code described by Efstathiou et al. (1985). The density on a regular grid is found by interpolating particle positions using the cloud–in–cell (CIC) method, and Poisson's equation is solved on this grid using Fast Fourier Transforms. As noted in the introduction, the FFT technique is highly efficient and scales as $N\log N$ (Hockney & Eastwood 1981). In order to obtain subgrid resolution a tree code is used. One possible manner of doing this is to compute shorter range forces for every particle with a tree code (Bagla 1999), in a manner analogous to the P^3M algorithm. Instead, TPM uses domain decomposition, taking advantage of the fact that potentials due to matter outside and inside a given domain can be added linearly, and it uses different solvers to find these two potentials. This section begins with a very broad description of the TPM algorithm, and then considers various aspects in more detail.

A basic outline of TPM can be summarized as follows:

1. Identify tree regions and the particles in each such region; particles outside of tree regions are evolved with PM only.
2. Push PM particles to midstep
3. Find the tidal potential in each tree region— the potential due to all mass outside of that region.
4. Integrate each tree region forward to the middle of the PM step.
5. Find the PM potential and update the acceleration for PM particles.
6. Integrate each tree region forward to the end of the PM step.
7. Update PM particle velocities and positions to the end of the time step.

The motivation behind this algorithm is that the motions of particles within a given tree region can be integrated independently using high temporal and spatial resolution: knowledge of the rest of the simulation is not required, except for the time–averaged, externally produced tidal forces.

The stepping in time of positions $\mathbf{x}$ and velocities $\mathbf{v}$ is accomplished with a standard second order leapfrog integration in comoving coordinates, the cosmological model determining the scale factor $a(t)$:

$$\mathbf{x}(t+\tfrac{1}{2}\Delta t) \;=\; \mathbf{x}(t) + \tfrac{1}{2}\mathbf{v}(t)\Delta t, \tag{1a}$$

$$\mathbf{g}(t+\tfrac{1}{2}\Delta t) = -\nabla\Phi(\mathbf{x}), \tag{1b}$$

$$\mathbf{v}(t+\Delta t) = \frac{1-H\Delta t}{1+H\Delta t}\mathbf{v}(t) + \frac{a^{-3}}{1+H\Delta t}\mathbf{g}(t+\tfrac{1}{2}\Delta t)\Delta t, \tag{1c}$$

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t+\tfrac{1}{2}\Delta t) + \tfrac{1}{2}\mathbf{v}(t+\Delta t)\Delta t. \tag{1d}$$

Here a, $H \equiv \dot{a}/a$, and the gravitational potential Φ are determined at time $t+\frac{1}{2}\Delta t$.

One advantage of the TPM approach is that it allows the use of multiple time steps. Tree particles are required to take at least two steps per PM step, but each particle has an individual time step so that finer time resolution can be used if required. Particles are arranged in an hierarchy of time bins differing by a factor of two, in the manner of Hernquist & Katz (1989):

$$\Delta t_{tree} = \frac{\Delta t_{PM}}{2^s}, \tag{2}$$

where Δt_{PM} is the PM time step and the integer $s \geq 1$. In a sense TPM operates along the same lines as a tree code, except that the particles in the $s = 0$ time step bin are handled differently from the rest. A diagrammatic representation of the time stepping is shown in Fig. 1 for the case when all tree particles are in the longest time step bin, $s = 1$. Beginning at time t, the PM particle positions are moved forward to midstep (eq. [1a] with $\Delta t = \Delta t_{PM}$). The tidal potential is then found, and the tree particles are evolved forward one full step with $\Delta t = \Delta t_{PM}/2$. The PM potential is updated (eq. [1b]), and for a second time the tree particles are evolved for one full step, to time $t + \Delta t_{PM}$. Finally, the PM particle velocities and positions are updated to the end of the step (eq. [1c-d]).

To repeat the TPM algorithm in more detail:

1. Identify tree regions and the particles in each region. This is done by identifying PM cells above a given density threshold, described in §3.1. Adjoining cells are then grouped into isolated tree regions, as described in BOX.

2. Push PM particles to midstep (eq. [1a]).

3. Find the tidal potential Φ_{ext} for each tree region. First the total potential is computed on the grid in the standard PM manner. Then for each of the tree regions, a small portion of the grid containing the PM potential is saved, and the contribution to this potential made by the tree itself is subtracted out, leaving the tidal potential due to all external mass. See BOX for details.

Since Φ_{ext} can only be calculated once per PM step, it will be calculated at midstep. PM particles are thus already in the proper location for this, but tree particles, with positions at the beginning of the PM time step, are not. However, since tree regions are spatially separated and the halo density profiles are evolving on slower timescales than any individual particle's orbital period,

an approximate position is sufficiently accurate. (Note that since a given tree's own contribution to Φ_{ext} is exactly subtracted back out, only the effect on other trees needs to be considered). Thus, an approximation of the tree particles' positions at the middle of the PM step is taken to be the position one full particle time step ahead; since tree particle time steps are half the PM time step or less, this is no more than the PM midstep. With these advanced positions, the potential on the grid is found in the standard PM manner.

For each tree the PM potential in a cubical subvolume is saved. This volume is slightly larger than the active cell region (in order to allow for finding the gradient of the potential through finite differencing) plus one extra cell on a side in case particles migrate out of the active cell region during the time integration. Since the PM time step is limited by a Courant condition (see §3.2) such that PM particles cannot move more than a fraction of a cell per step, one extra cell is sufficient— particles near the edge of a tree region are at only a slightly higher overdensity than the densest PM regions, and hence have similar velocities. At the beginning of a tree step the portion of the PM potential due to the tree particles themselves is subtracted out, leaving the tidal potential. With Φ_{ext} saved in this manner, every time a particle's acceleration is updated, the tidal force is calculated from the grid in the same manner that PM forces are found. This is an improvement over the method used in BOX.

4. Integrate each tree region forward to the middle of the PM step. This is done with a tree code, adding in the tidal forces. The tree code we use was written by Lars Hernquist (Hernquist 1987; Hernquist & Katz 1989; Hernquist 1990). Any other potential solver could be used, but a tree code is well suited for this type of problem, in that it can efficiently handle a wide variety of particle distributions, can include individual particle time steps, and scales as $N\log N$.

Note that for each tree this portion of the code is self–contained; that is, once the particle data and potential mesh for a given tree have been collected together no further information is required to evolve that tree forward. This makes TPM well suited for parallel processing on distributed memory systems. Once the tree data is received by a given processor this step— which is the most computationally expensive part of the algorithm— can be performed without any further communication. Given this coarse–grained parallelism, one can use widely distributed and heterogeneously configured processors, with more capable processors (or groups of processors) reserved for the largest trees.

5. Find the density and potential on the PM grid again, and update the acceleration for PM particles (eq. [1b]). All tree particles are at the PM midstep, the proper location for computing the force on PM particles.

6. Integrate each tree region forward to the end of the PM step.

7. Update PM particle velocities and positions to the end of the time step (eqs. [1c-d]).

3. Other Aspects of the Implementation

3.1. Domain Decomposition

TPM exploits the fact that gravitational instability in an expanding universe creates spatially isolated high density peaks; for typical CDM models these peaks will contain a significant fraction of the mass ($\sim 1/4$) but occupy a tiny fraction of the volume ($\sim 10^{-3}$; see for example Fig. 2 of Mo & White (2002)). As described in detail in BOX, this is accomplished by tagging active mesh cells based on the cell density (which has already been calculated in order to perform the PM step). Adjacent cells are linked together, dividing the cells into groups in a friends–of–friends manner. Thus isolated high–density regions of space are identified, separated by at least one cell width.

In BOX, the criterion used to decide if a cell i was active was if its density ρ_i exceeded a global threshold density

$$\rho_i \geq \rho_{\rm thr} = A\bar{\rho} + B\sigma, \tag{3}$$

where $\bar{\rho}$ is the mean density and σ is the dispersion of cell densities. While temporally adaptive (in the sense that the threshold is lower at early times when peaks are rare), this criterion has two main drawbacks. First, small peaks in isolated regions or voids will not be picked up, especially when σ is large, because the total mass in such a halo would not put a PM cell above $\rho_{\rm thr}$. Secondly, in the case of a region with peak density near $\rho_{\rm thr}$, whether or not a given PM cell is above $\rho_{\rm thr}$ can depend on the offset of the grid with respect to the particles— the halo mass may be divided into two or more cells.

We have developed an improved criterion for selecting tree regions involving the contrast of a cell with its surroundings (rather than a single global threshold value) found by comparing the density smoothed on two scales. For each cell three densities are computed. First ρ_s, found by boxcar smoothing over a length of 5 cells along each dimension. (Another type of smoothing, *e.g.* with a Gaussian, would be possible, but the boxcar can be done with minimal interprocessor communication). Second, ρ_c, the mean density in the eight–cell cube of which the cell under consideration is the lower octant (when using CIC, a particle in this cell would also assign mass to the other cells in this cube). Third, $\rho_{\rm ext}$, a measure of the density surrounding this cube, defined as

$$\rho_{\rm ext} = \frac{5^3\rho_s - 2^3\rho_c}{5^3 - 2^3}. \tag{4}$$

All eight cells used to find ρ_c are marked as active if

$$\rho_c \geq \rho_{\rm thr} = (A\bar{\rho} + B\sigma)\,\frac{\rho_{\rm ext}}{\bar{\rho} + \sigma + \rho_{\rm ext}}. \tag{5}$$

The reasons for such a choice of form can be seen by examining the behavior of eq. [5] in various limits. At early times in a cosmological simulation $\rho \approx \bar{\rho}$ and $\sigma \ll \bar{\rho}$, so $\rho_{\rm thr} \approx \frac{1}{2}A\rho_{\rm ext} \approx \frac{1}{2}A\bar{\rho}$. Thus

with a choice of $A = 2$ those regions which are only slightly overdense will be selected. At late times $\sigma \gg \bar{\rho}$. In void regions $\rho_{\rm ext} \ll \sigma$, which means $\rho_{\rm thr} \approx B\rho_{\rm ext}$; thus an isolated cell will be chosen if it is a fixed multiple above its surroundings. If on the other hand $\rho_{\rm ext} \gg \sigma$, then $\rho_{\rm thr} \approx B\sigma$, meaning that at each step there is a given density above which all cells will be chosen; this limit will increase with time as σ increases. Keep in mind that the distribution of cell densities will be approximately lognormal (Kayo, Taruya & Suto (2001) and references therein) when interpreting the value of σ.

After cell selection, the domain decomposition proceeds in the manner described in BOX. Cells are linked by friends–of–friends, yielding regions of space separated by at least one cell length. Any particle which contributes some mass to an active cell when finding the PM density is assigned to the corresponding tree.

Because the volume of all tree regions is less than one percent of the total volume, the amount of memory allocated to hold the tidal potential data is generally not significant compared to that already needed for the PM mesh. However, one complication of this scheme is that at early times, when σ is small, the resulting trees can be very long filaments. While not containing a large amount of matter, being not very overdense and nearly one–dimensional, the size of a cubical volume enclosing such a tree will sometimes be a significant fraction of the entire simulation volume. This can cause difficulties because of the amount of computer memory required to compute and save the tidal potential in such subvolumes. The value of B in eq. [5] is thus allowed to increase at early times when the spatial extent of trees tends to become larger, and decrease at later times when trees are more compact. We have settled on the following method, which has worked well in a variety of simulations. Space is allocated for a maximum subvolume length one quarter of the PM mesh size. The size of the largest sub-box used is checked at the end of each PM step. At early times (when $\sigma < 1.6$) B is increased by 0.5% if this box has a length more than a third of the allocated value. Once σ exceeds 1.6 in a typical simulation, the spatial extent of trees has stopped growing, so in this case B is decreased by 0.25% if the largest sub-box has a length less than half the maximum allocated. A minimum value is set for B to prevent too many tree particles being chosen; a choice of $B = 10$ will place roughly half of the particles in trees at $z = 0$. An example of this in practice is discussed in §3.3.

The number of PM grid cells can be more or less than the number of particles, though of course the finer the grid, the greater the accuracy of the PM and tidal forces. Generally, the number of cells should at least equal the number of particles; we have found that eight grid cells per particle works well.

3.2. Time Step Criterion

A variety of methods for determining the particle time step have been proposed. This diversity of criteria reflects the fact that the appropriate time step can depend on a number of quite different

considerations. These include the local density or the local dynamical time, nearby substructure, the nearest neighbor, and the softening length (Knebe et al. 2000; Power et al. 2002). In addition, for any single criterion, one can imagine a special circumstance where it breaks down. This suggests the possibility of basing Δt on more than one criterion. Thus in the TPM code we have decided to combine a number of possible criteria, drawn from variables which do not require any significant computation to find, in the following manner:

$$\Delta t = \eta \cdot \mathrm{MAX}\left(\Delta t_1,\, \Delta t_2\right), \text{where} \tag{6a}$$

$$\Delta t_1 = \mathrm{MIN}\left(\epsilon/v,\, \sqrt{a^3\epsilon/g}\right), \text{and} \tag{6b}$$

$$\Delta t_2 = \mathrm{MIN}\left(v/g,\, g/\dot{g},\, \frac{\epsilon^2 + r^2}{rv}\right). \tag{6c}$$

Here η is an adjustable dimensionless parameter, ϵ is the spline kernel softening length, and r is the distance to the nearest neighboring particle. The latter is found as a byproduct of the force calculation; if at the beginning of a tree step r has not been calculated, one can as a proxy find the minimum r that is consistent with the particle's current Δt. The time derivative of g is approximated by $\dot{g} = |g(t) - g(t - \Delta t_s)|/\Delta t_s$. This criterion for Δt is quite conservative, and is similar in principle to that adopted by Aarseth (1999).

A comparison of the time step returned by eq. [6] and the often used criterion $\eta\sqrt{\epsilon/g}$ is given in Fig. 2, which shows how these two criteria compare at $z = 0$ for particles in the largest tree of the simulation discussed in §3.3 and §4.2. This tree is made up of over 1.4×10^5 particles, and contains two large halos plus a number of smaller satellites and infalling matter. Each contour level in Fig. 2 encloses a tenth of the particles, and the remaining tenth are plotted as points. It can be seen that eq. [6] tends to yield a smaller timestep than $\eta\sqrt{\epsilon/g}$; this is the case for 74% of the particles. Generally the differences are not large— less than a factor of two for 60% of the particles. Another trend seen in the figure is that as $\eta\sqrt{\epsilon/g}$ becomes smaller it is more likely for eq. [6] to give an even smaller value, and conversely eq. [6] tends to give a longer time step than $\eta\sqrt{\epsilon/g}$ as the latter becomes larger. In most cases the value of Δt is set by the factor $(\epsilon^2 + r^2)/(rv) = (r/v)(1 + \epsilon^2/r^2)$. In other words, the distance to the nearest neighbor is not allowed to change by a large factor, unless this distance is small compared to the softening length. Even in cases when this factor is larger than $\sqrt{\epsilon/g}$, it is still smaller than v/g and $g/\dot{g}$; since the equations being integrated have the form $\Delta x = v\Delta t$ and $\Delta v = g\Delta t$ this limits integration errors.

The PM time step is limited by a Courant condition, such that no PM particle moves more than one quarter of a cell size per step. Also, a is not allowed to change by more than 1% per step. Initially this latter criterion is the most restrictive, but over time it allows a longer Δt_{PM} and becomes unimportant. The Δt_{PM} allowed by the Courant condition also tends to increase over time, although more slowly. This is because particles falling into dense regions are placed into trees, and the velocities of the remaining particles are redshifted (eq. [1c]). The largest time step for any particle, Δt_{PM}, is kept constant until it can safely be increased by a factor of 2, and it is then doubled. The time step for tree particles is unchanged, which means they take twice as many

steps per PM step as before; during tree evolution particles will move into a lower time step bin if allowed by eq. [6]. Whenever any particle changes time step, its position is updated as described in Hernquist & Katz (1989) to preserve second order accuracy.

3.3. A typical run

Various aspects of a typical TPM run can be demonstrated with a standard cosmological simulation. This test case contains N=128^3 particles in a box L=$40h^{-1}$Mpc on a side. The number of grid points for the PM and domain decomposition portions of the code is eight times the number of particles, or 256^3. The initial conditions were generated using the publicly available codes GRAFIC1 and LINGERS[1] (Bertschinger 2001; Ma & Bertschinger 1995). A spatially flat LCDM model was chosen, with cosmological parameters close to the concordance model of Ostriker & Steinhardt (1995): $\Omega_m = 0.3, \Omega_\Lambda = 0.7, h = 0.70, \Omega_b h^2 = 0.20, n = 1$, and $\sigma_8 = 0.95$. The particle mass is thus $2.54 \times 10^9 h^{-1} M_\odot$. Within the tree portion of the code, the opening parameter θ in the standard Barnes–Hut algorithm (Barnes & Hut 1986) is set to $\theta = 0.577 \approx 1/\sqrt{3}$, and the time step parameter η=0.3 (see eq. [6]). The cubic spline softening length was chosen to be $3.2h^{-1}$kpc so that the spatial dynamic range $L/\epsilon \sim 10^4$. With such a small softening length, halos with fewer than 150 particles are likely to undergo some 2-body relaxation in their cores over a Hubble time (assuming the particle distribution follows an NFW profile with concentration c=12).

Fig. 3 displays the evolution of several important quantities during this run as a function of expansion parameter a. The top panel shows the dispersion of PM cell densities σ (where the mean cell density is unity), and the second panel shows the value of B; both of these factor in eq. [5], which in turn plays a major part in determining the tree distribution. Various aspects of this distribution are shown in the remaining curves. The third panel shows the size of the largest cubical subvolume required (in units of PM grid cells). Initially small, this grows extremely rapidly as σ rises from its initial value of 0.3 to ~ 1; in response B is increased. When this size is at its greatest (at $\sigma \approx 1.6$), the percentages of total volume and mass in trees (shown in the next two panels) are still quite small. The trees at this time tend to follow caustics— they are only slightly overdense and not very massive, but because of their filamentary nature they can have a large spatial extent. These caustics then fragment and collapse, so— even while the total mass in trees and the number of trees (shown in the third panel from the bottom) increase— the maximum subvolume size decreases. The changes in B affect mainly the maximum subvolume size; the number of trees and volume contained in tree regions both increase at a steady rate up to z=1. The number of tree particles increases monotonically throughout the simulation.

After $z = 1$ (by far the bulk of the computational time), the characteristics of the simulation change much more slowly. Roughly half the particles are in ∼2700 trees ranging from a few to

[1]These codes are available at http://arcturus.mit.edu/grafic/

10^5 particles but occupying only 0.4% of the simulation volume. The penultimate panel shows the number of particles in the first and third largest trees; by $z = 1$ the growth in mass of these objects has slowed to a fairly constant, low rate. Trees are distributed in mass roughly as a power law, with 67% of the trees having fewer than 100 particles and 95% less than 1000. Most tree particles take two or four steps per PM step, but some are taking up to 64; there were 1075 PM steps total in this run.

As σ increases, so does the limiting density above which all cells are treated at full resolution. It is important to keep track of this limit when interpreting the results of a TPM run. The bottom panel shows the highest value of the density among those cells evolved only with PM. By $z = 1$ this density is 120 times the mean, corresponding to 15 particles inside a cell. By the end of the computation, the densest PM–only cell contains 26 particles; so trying to make statements about objects smaller than this will be complicated by the varying spatial resolution of TPM. Note that most objects with $N \lesssim 25$ will in fact be followed at full resolution— substructures inside larger objects because they are in regions of higher density, and small isolated halos because they present a density contrast to their surroundings. To be cautious we will limit our analysis to objects 50% larger than this limit, *i.e.* 40 particles or $10^{11}h^{-1}M_\odot$.

3.4. Load balancing

As seen in the previous section, during later epochs— which take up most of the computational time needed for a run— the mass distribution of trees is generally well fit by a power law ranging from a few particles up to of order $0.01N$. The actual mass of the largest tree will depend on the ratio of the largest non-linear scale in the box to the box size; as this ratio becomes larger so does the mass of the largest tree. Furthermore, the amount of computation required for a tree with N_t particles will scale as $N_t \log N_t$. Thus, there can be a substantial variation in the computational time required between different trees, and evolving the largest tree can comprise a significant fraction of the total computational load. An efficient parallel code must handle this situation well when dividing work up among NCPU processors.

Load balancing of trees is achieved in three ways. First, trees are sorted by N_t and then divided into "copses" of roughly equal amounts of work using the "greedy" algorithm. That is, starting with the largest tree, each one in turn is given to the copse which up to that point has been assigned the least amount of total work. Usually there is one copse per CPU, but there can be two or more per CPU if required by space constraints. There is a communication step, when all data associated with a copse is sent to one CPU. A CPU then evolves each tree in its local copse in turn, starting with the most massive. Additionally, the largest trees can be done in parallel. If a few large trees dominate the work load, then it is impossible for all copses to contain equal amounts of work— ideally, one would want NCPU copses, each comprising 1/NCPU of the total amount of work, but this clearly can't happen if the largest tree takes a larger fraction just by itself. In this case, the force calculation for these large trees is done in parallel by a small number of CPUs. This

is currently done quite crudely, with only the tree walk and force calculation actually carried out in parallel; in theory a fully parallel tree code could be used for every tree, allocating more processors to those copses containing the most work. Only a few nodes are usually required to reduce the time spent on the largest tree to the level required for load balancing. As a final means of balancing the load, when a CPU finishes evolving all the trees in its local copse, it then sends a signal to the other CPUs. A CPU with work still left will send an unevolved tree to the idle CPU for it to carry out the evolution.

The scaling of the current implementation of TPM with number of processors is shown in Fig. 4. The test case for these timing runs is a standard LCDM model at redshift z=0.16 with N=256^3 particles in a 320h^{-1}Mpc cube. This particular set of runs was carried out on an IA-64 Linux cluster at NCSA named "Titan". This machine consists of 128 nodes, each with dual Intel 800MHz Itanium processors, and a Myrinet network interconnect. While the total time required depends on processor performance, similar scaling with NCPU has been found on a number of machines with various types of processors and interconnects. The topmost line in Fig. 4 is total time, calculated as the number of seconds wallclock time per PM step multiplied by NCPU; perfect scaling would be a horizontal line. TPM performs quite well— at NCPU=64 the efficiency is still 91% as compared to NCPU=4. Beyond this point scaling begins to degrade, with the efficiency dropping to 75% for NCPU=128.

The reason TPM scales well can be seen in the second line from the top, which shows the total time spent in tree evolution. This part of the code takes up most of the CPU time, but it requires no communication and there are enough trees so that just coarse–grained parallelization works reasonably well. The next two curves shown indicate the amount of time in the PM portion of the code and overhead related to trees (identifying tree regions and particles, etc.). These take a small fraction of the total time and scale well since the grid is distributed across all processors. The final curve in Fig. 4 shows the time related to communication and imbalance in the tree part of the code. As NCPU increases it becomes more difficult to divide the tree work evenly and processors spend more time either recruiting work from others or waiting for them to finish. This overhead could likely be reduced by incorporating a fully parallel tree code and also by computing the nonperiodic FFT (required to find the tidal potential— see BOX) in parallel. A run with a larger number of particles and grid points, and thus a larger number of trees, would show efficient scaling beyond NCPU=128.

4. Performance and Accuracy

4.1. Spherical Overdensity Test

As a test of new code we have carried out a simulation of the secondary infall and accretion onto an initially uniform overdensity. This can be compared both to other codes and to the analytic, self-similar solution of Bertschinger (1985). To create the initial conditions, 64^3 particles were

placed on a uniform grid with zero velocity. The eight particles on the grid corners were then removed and placed at the center of the volume with a spacing one half that of the regular grid. Thus, this is actually more of a cubic overdensity than spherical, but it quickly collapses and the subsequent infall is independent of the details of the initial state. Also, there is a void located half the box distance away from the overdensity, but the evolution is not carried out for a long enough time for this to be significant. The initial condition is integrated from expansion factor $a = 0.01$ to $a = 1$. The values $A = 2$ and $B = 10$ were used for the constants in eq. [5]. Since only one halo is forming in the box, σ remains small, rising to only 0.5 by the end of the run. At the end, there are roughly 500 particles within the turnaround radius. This test can be seen as exploring how well the initial collapse of small objects is followed, which is the beginning stage of halo formation in an hierarchical scenario.

The final state of this run is shown in Fig. 5. The top panel shows density as a function of radius. Filled points are the TPM model; error bars are the square root of the number of particles in a given radial bin. The inner edge of the innermost bin is twice the softening length. Also shown is the same run carried out with a P^{3}M code (as open circles; details of this code are discussed in §4.2), and the solution of Bertschinger (1985). The agreement is quite good— the main limitation of this run is likely the mass resolution. As a measure of the phase space density, the lower panel shows $\rho/\langle v\rangle^3$, where $\langle v\rangle$ is the velocity dispersion of the particles in each radial bin. Following Taylor & Navarro (2001), we compare this to the Bertschinger (1985) solution for a $\gamma = 5/3$ gas. Again the agreement is quite reasonable. Differences between TPM and P^{3}M arise from different types of softening (P^{3}M uses Plummer softening; here ϵ was set to half the TPM value) and from time stepping (for P^{3}M, Δt for all particles was set by the minimum $\sqrt{0.05\epsilon/g}$).

4.2. Comparison with other codes

As a test of TPM in a less idealized situation, the initial conditions described in §3.3 were again evolved, but using two other N-body codes. The first is the P^{3}M code of Ferrell & Bertschinger (1994) in a version made parallel by Frederic (1997)— this code was also used in §4.1. As in the TPM run, a mesh of 256^3 grid points was used. This code uses Plummer softening; the softening length was set to half the value used in the TPM run. The time variable in this code is τ, defined by $d\tau = dt/a^2(t)$; all particles have the same time step, set by the minimum $\sqrt{0.05\epsilon/g}$. The other N-body code used is the tree code named GADGET[2] of Springel, Yoshida & White (2001). In this code, periodic boundary conditions are handled with Ewald summation, the time variable is the expansion factor a, and each particle has an individual time step. Following Power et al. (2002), the conservative tree node opening criterion flagged by -DBMAX was used, and the time step was set by $\sqrt{0.03\epsilon/g}$. In addition, the maximum allowed time step was set to 1% of the initial expansion factor; whenever a doubled this maximum was also increased by a factor of two, by checkpointing

[2]Publically available at http://www.MPA-Garching.MPG.DE/gadget/

the run and restarting with the new value. The softening length was set to be the same as the TPM run (allowing for different notational conventions).

As a first comparison of the codes, the two point correlation function $\xi(r)$, found by counting the number of particle pairs in bins of separation r, was calculated. To compute $\xi(r)$, 51 logarithmically spaced bins were used, with the minimum pair separation considered being $2\epsilon = 6.4h^{-1}$kpc and the maximum one third of the box size. The results at various redshifts are shown for all three runs in Fig. 6. Clearly there is little difference to be seen in the three codes. At larger scales this is to be expected; at smaller scales force approximations, smoothing, and relaxation may become important. The fact that P^{3}M uses Plummer rather than spline softening explains why it shows a lower $\xi(r)$ at scales less than a few times the softening length. If high values of $\xi(r)$ provide a measure of accuracy, the TPM and GADGET are slightly more accurate than P^{3}M.

The statistics of mass peaks— the dark halos surrounding galaxies, clusters, and so on— are an important product of N-body codes (White 2002). One commonly used halo finder is friends–of–friends, or FOF[3] (Davis et al. 1985). The cumulative mass function, found with FOF using a linking length of 0.2 times the mean interparticle separation, is shown in Fig. 7. At higher redshifts there is little discernible difference between the three codes. At later times, TPM seems to have fewer halos containing $\lesssim$50 particles. It is unclear how to interpret this, because halos with less than 150 particles (i.e. mass $< 3.8 \times 10^{11} h^{-1} M_\odot$) are affected by two-body relaxation. Knebe, Green & Binney (2001) found that an adaptive P^{3}M and GADGET both produced more small FOF halos than the adaptive multigrid MLAPM and ART codes, so in this case TPM may agree more closely with the latter.

To investigate further a different halo finder was used, namely the Bound Density Maxima (BDM) algorithm[4] (Klypin et al. 1999). This algorithm is significantly different from FOF, so the resulting mass function is probing different qualities of the dark matter distribution. Density maxima inside spheres of radius 100 h^{-1}kpc were found, and halo centers were then calculated using spheres of radius 40 h^{-1}kpc. The mass for each halo is taken to be that inside a sphere containing the overdensity expected for a virialized halo just collapsed from a spherical top–hat; to calculate this overdensity the fit of Bryan & Norman (1998) was used. In BDM (unlike FOF) particles moving faster than the escape velocity are removed from the halo. The BDM cumulative mass functions for the three N-body codes are shown in Fig. 8. As with FOF, the agreement between codes is quite good. In this case it appears to be TPM which produces more small halos, but again this effect is for low mass halos likely affected by relaxation.

Using the BDM halos with more than 40 particles, the halo–halo correlation function is shown in Fig. 9. The number of pairs of halos in 21 logarithmically spaced radial bins, with the smallest

[3]We employed the University of Washington NASA HPCC ESS group's FOF code, available at http://www-hpcc.astro.washington.edu/

[4]Publically available at http://astro.nmsu.edu/∼aklypin/

separation being 200 h^{-1}kpc and the largest one third of the box size, were tallied and compared with the expectation for a random distribution. All three codes give the same result. From the results presented so far in this section it is clear that TPM yields a quite similar evolved matter distribution as compared to other codes.

Finally we turn to the internal properties of halos. One probe of the mass distribution in a halo is the circular velocity $v_c \equiv \sqrt{GM(<r)/r}$. For each simulation we divide halos up into six mass bins. To avoid relaxation effects, the lowest mass considered is $3.81 \times 10^{11} h^{-1} M_\odot$. The width of each bin is a factor of $\sqrt{10}$, so the largest mass bin contains two halos with mass above $1.20 \times 10^{14} h^{-1} M_\odot$. For each set of halos the average circular velocity as a function of radius was calculated; these average velocity curves are shown in Fig. 10. It can be seen that these curves are quite similar for all three codes. Two main trends are noticeable, the main one being that P^{3}M shows lower circular velocities at small radii. This is likely due to the use of Plummer softening and less strict time step criterion in the P^{3}M run, both of which would lead to lower densities in the innermost parts of halos. The other main difference is that for the lowest masses it appears that TPM yields higher circular velocities than the other two codes.

Instead of averaging over a number of halos of similar mass, it is also possible to compare halos on a one-by-one basis. To find the appropriate pairs of halos, the list of halos with more than 150 particles found by BDM was sorted by mass, and for each target GADGET or P^{3}M halo a TPM halo was selected as a match if its mass was within 25% and its position within 1 h^{-1}Mpc of the target halo; if more than one halo passed this test then the nearest in position was selected. The selected TPM halo was removed from further consideration, and the process repeated for next target halo. In this manner a match was found for 588 out of 609 GADGET halos and 573 out of 582 P^{3}M halos. For various measured halo properties, the percentage difference of the TPM halo from the GADGET or P^{3}M halo was calculated. Since the dispersion of these differences increases as less massive halos are considered, halo pairs are split into two groups, with the higher mass group containing all target halos with 394 or more particles (i.e. mass $\geq 10^{12} h^{-1} M_\odot$).

Results are shown in Table 1, which for each property gives the mean percentage difference and one standard deviation, as well as the first quartile, median percentage difference, and third quartile. The first property shown is halo mass M. The difference here is constrained to be less than 25%, but in most cases the TPM value is within 10% of the other code's. It appears that P^{3}M yields slightly lower masses than the other two codes, but the latter two agree quite well. The agreement in masses shows that the simple scheme used to find matching halo pairs works well, as does the fact that the difference in position is less than $165 h^{-1}$kpc in 95% of the cases. This is reinforced by the agreement between the halo center of mass velocities v_{cm} shown in Table 1; the velocity vectors are closely aligned, differing by less than 7.5° in 95% of the pairs. Thus only a few percent of the halo pairs are mismatches; since these pairs still have similar masses and are likely in similar environments, they are kept in the comparisons.

In terms of the 3-D root mean square velocity v_{rms} and maximum circular velocity v_c, the

more massive halos are very similar in all three codes, with no offset between codes. However, at the lower mass end TPM gives values that tend to be systematically higher by a few percent. This can also be seen for the average circular velocity in the bottom curve of Fig. 10; this is curve is the average for halos below roughly $10^{12}h^{-1}M_{\odot}$.

The difference between TPM and the other two codes is most pronounced near halo centers. The last property shown in Table 1 is a comparison of central density ρ_o, which is computed by measuring the amount of mass within 4ϵ=$12.8h^{-1}$kpc of the halo center. As would be expected from Fig. 10, TPM yields higher central densities than GADGET, with P^{3}M giving lower ρ_o than either of these.

One possible source of the differences seen in TPM halos as compared to a pure tree code could simply be the choice of time step. The P^{3}M run took 2.1×10^4 steps and the GADGET run 2.8×10^4; TPM on the other hand took 4.4×10^4, or 60% more than GADGET. Of course this comparison is not entirely straightforward because different particles determine the smallest time step at different times. Part of the difference between GADGET and TPM may be due to the fact that TPM bins time steps by factors of two, whereas GADGET allows time steps to vary more gradually. A second TPM run was carried out to separate other code differences from the time step criterion. This run was identical to the first except that the time step was set not by eq. [6], but rather by $\sqrt{0.05a^3\epsilon/g}$ (similar to the GADGET and P^{3}M codes); as a result it took fewer steps than any of the other runs. There is no significant difference in the mass functions and halo–halo correlation functions between this run and the original TPM run. The particle-particle correlation function is different, however. This can be seen in Fig. 11, which shows the ratio of the original TPM run's $\xi(r)$ to that of the new run, at redshift z=0. The new TPM run has a lower $\xi(r)$ for $r < 20h^{-1}$kpc, roughly 5-10% lower than the original TPM and similar to the P^{3}M run. This indicates the internal structure of halos has been affected by the longer time steps, which is confirmed by repeating the comparison of individual halos. The difference between halos in the new TPM run and the GADGET run are given in Table 1. For the higher mass halos the new run, unlike the original one, tends to give lower maximum circular velocities and central densities than the tree code. This indicates that the longer time step is in fact leading to inaccuracies. The differences between TPM and the tree code for low mass halos seen in the original run persist in the new run, though they are not quite as pronounced.

Another point of comparison between codes is efficiency with which they use computing resources. Fig. 12 shows the wall-clock time consumed by each of the codes, as a function of expansion parameter, in carrying out the test simulation. All the runs used four 300 MHZ IP27 processors of an SGI Origin 2000; the codes were compiled with the SGI compilers and MPI library. All three codes used roughly the same amount of memory (TPM requires at least 20 reals per particle plus 3 reals per mesh point, divided evenly among processors). At the earliest times both P^{3}M and TPM spend most of their time in the PM FFT, and so they behave similarly. However, as objects begin to collapse P^{3}M begins to consume more time computing particle-particle interactions. The fact that the accelerations of all particles are updated every step also makes P^{3}M use more time than

do the two multiple time step codes. The tree code takes more time when the particle distribution is nearly homogeneous, demonstrating that a PM code (which is what the gridded codes basically are in this situation) is very efficient. However, the tree code timing is roughly independent of the particle distribution, and once inhomogeneity develops it does not require more time per update, whereas the other codes do. TPM does well compared to the tree code for a couple of reasons. By a=1 roughly half the particles in the TPM run are still being handled solely by PM. Also, imagine breaking N particles up into t trees each with N/t particles; the time required to solve all the trees will scale as $N \log N/t$, lower by a logarithmic factor than using the same tree solver on all the particles. Overall, despite taking more timesteps, in this test case TPM required less CPU time than the other codes, by a factor of ~ 4 relative to the tree code and ~ 8 relative to P^{3}M.

5. Conclusions

This paper has presented a parallel implementation of the TPM algorithm. Several improvements over the implementation of BOX have been made. Particles in tree regions each have an individual time step, half of the PM time step or less, making the tree integration more efficient. The treatment of tidal forces on trees is also improved by saving the tidal potential on a grid and evaluating the force as a function of particle position at each smaller particle time step; thus a greater ratio of tree to PM time steps is allowed. A new, more stringent, time step criterion has been implemented. A new criterion for locating regions for treatment with higher resolution is given; by finding cells with higher density than their surroundings, small halos in lower density regions are located and followed at full resolution. These changes significantly increase the speed and accuracy of TPM: a computation of the test case discussed in §4.2 using the code described in BOX required more CPU time (by a factor of over three) than the current version, but was only accurate for halos with more than 315 particles.

Comparisons with other widely used algorithms were made for a typical cosmological structure formation simulation. These show excellent agreement. The particle-particle correlation functions, the halo mass functions, and the halo-halo correlation functions from TPM agree quite well with those from P^{3}M and tree codes. The internal properties of halos also agree; the main difference being that, for lower mass halos, TPM yields higher rms and maximum circular velocities (by a few percent over a tree code). TPM halos also show higher central densities than those of the other two codes, though the mean difference is smaller than the dispersion. This difference disappeared, at least among the more massive halos, when the time step criterion of eq. [6] was replaced with one similar to that employed by the other two codes. Thus we conclude that a choice of a relatively conservative time step criterion contributed to a slightly improved accuracy.

TPM yielded results of similar accuracy to the other codes used here while using significantly less computational time (about a quarter of that needed by a tree code and an eighth of that needed by P^{3}M). It also scales well on distributed memory parallel machines, such as networked PCs, because this parallelism is built in as part of the design of TPM. However, in committing to

an algorithm which accentuates the coarse-grained parallelism inherent in a typical cosmological simulation, a large degree of flexibility is sacrificed. A basic presumption of TPM is that the largest nonlinear structure inside the simulation box is a small fraction of the total mass and volume. To simulate a situation where this is not the case (e.g. two colliding galaxies) another code would be preferred.

The TPM source code can be obtained at http://astro.princeton.edu/∼bode/TPM or by contacting the authors. The code is written in Fortran 77 and uses MPI for message passing; thus it is very portable and can be used on clustered PC's or other distributed memory systems.

Many thanks are due to Lars Hernquist for generously supplying a copy of his tree code; also Edmund Bertschinger for use of his P^3M code, Joe Henawi for help with FOF, and Scott Tremaine for useful discussions. This research was supported by the National Computational Science Alliance under NSF Cooperative Agreement ASC97-40300, PACI Subaward 766. Computer time was provided by NCSA and the Pittsburgh Supercomputing Center.

REFERENCES

Aarseth, S. 1999, PASP, 111, 1333
Baertschiger, T., Joyce, M. & Labini, F.S. 2002, ApJ, 581, 63 (astro-ph/0203087)
Bagla, J.S. 1999, preprint (astro-ph/9911025)
Barnes, J.E. 1998, Galaxies: Interactions and Induced Star formation, R.C. Kennicutt Jr., F. Schweizer and J.E. Barnes, Berlin: Springer, 275
Barnes, J. & Hut, P. 1986, Nature, 324, 446
Becciani, U. & Antonuccio-Delogu, V. 2001, Comp. Phys. Comm., 136, 54
Bertschinger, E. 1985, ApJS, 58, 39
Bertschinger, E. 1998, ARA&A, 36, 599
Bertschinger, E. 2001, ApJS, 137, 1
Bryan, G.L. & Norman, M.L. 1998, ApJ, 495, 80
Bode, P., Ostriker, J.P., & Xu, G. 2000, ApJS, 128, 561 (BOX)
Calder, A.C. et al. 2002, ApJS, in press
Couchman, H.M.P., Thomas, P.A. & Pearce, F.R. 1995, ApJ, 452, 797
Davis, M., Efstathiou G., Frenk, C. & White, S.D.M. 1985, ApJ, 292, 371
Dehnen, W. 2000, ApJ, 536, L39
Dehnen, W. 2002, J. Comp. Phys., 179, 27
Dorband, E.N., Hemsendorf, M. & Merritt, D. 2002, J. Comp. Phys., in press (astro-ph/0112092)
Efstathiou G., Davis, M., Frenk, C. & White, S. 1985, ApJS, 57, 241
Ferrell, R. & Bertschinger, E. 1994, Int. J. Mod. Phys. C, 5, 933
Frederic, J.J. 1997, Ph.D. Thesis, MIT
Hamana, T., Yoshida, N. & Suto, Y. 2002, ApJ, 568, 455
Hernquist, L. 1987, ApJS, 64, 715
Hernquist, L. 1990, J. Comp. Phys., 87, 137

Hernquist, L. & Katz, N. 1989, ApJS, 70, 419
Hockney, R.W. & Eastwood, J.W. 1981, Computer Simulation Using Particles, New York: McGraw Hill
Kayo, I., Taruya, A. & Suto, Y. 2001, ApJ, 561, 22
Jing, Y.P. & Suto, Y. 2002, ApJ, 574, in press
Klypin, A. 2000, preprint (astro-ph/0005502)
Klypin, A., Gottlöber, S., Kravtsov, A.V. & Khokhlov, A.M. 1999, ApJ, 516, 530
Klypin, A., & Holtzman, J. 1997, preprint (astro-ph/9712217)
Knebe, A. 2002, MNRAS, submitted (astro-ph/0201490)
Knebe, A., Green, A. & Binney, J. 2001, MNRAS, 325, 845
Knebe, A., Kravtsov, A.V., Gottlöber, S. & Klypin, A.A. 2000, MNRAS, 317, 630
Kravtsov, A.V., Klypin, A.A. & Khokhlov A.M. 1997, ApJS, 111, 73
Lia, C. & Carraro, G. 2001, Ap&SS, 276, 1049
Ma, C.-P., & Bertschinger, E. 1995, ApJ, 455, 7
Miocchi, P. & Capuzzo-Dolcetta, R. 2002, A&A, 382, 758
Mo, H.J. & White, S.D.M. 2002, MNRAS, 336, 112 (astro-ph/0202393)
Ostriker, J.P. & Steinhardt, P.J. 1995, Nature, 377, 600
Power, C., Navarro, J.F., Jenkins, A., Frenk, C.S., White, S.D.M., Springel, V. Stadel, J. & Quinn, T. 2002, MNRAS, submitted (astro-ph/0201544)
Ricker, P.M., Dodelson, S. & Lamb, D.Q. 2000, ApJ, 536, 122
Springel, V., Yoshida, N. & White, S.D.M. 2001, New Astronomy, 6, 79
Stadel, J.G. 2002, Ph.D. Thesis, University of Washington, Seattle
Taylor, J.E. & Navarro, J.F. 2001, ApJ, 563, 483
Teuben, P. 1995, Astronomical Data Analysis Software and Systems IV, R.A. Shaw, H.E. Payne & J.J.E. Hayes, San Francisco: Astronomical Society of the Pacific, 398
van Kampen, E. 2000, MNRAS, submitted (astro-ph/0002027)
Viturro, H.R. & Carpintero, D.D. 2000, A&AS, 142, 157
Xu, G. 1995, ApJS, 98, 355
White, M. 2002, ApJS, in press
Yahagi, H., Mori, M. & Yoshii, Y. 1999, ApJS, 124, 1
Yahagi, H. & Yoshii, Y. 2001, ApJ, 558, 463

AAS LaTeX macros v5.0.

Table 1: Halo comparison — percentage differences

	$3.8 \times 10^{11} < M < 10^{12}$					$M \geq 10^{12}\ h^{-1}M_\odot$				
	mean	s.d.	Q_1	Q_2	Q_3	mean	s.d.	Q_1	Q_2	Q_3
TPM vs. Tree										
M	-0.03	8.21	-5.07	0.08	4.80	-0.37	5.33	-2.33	-0.12	2.08
v_{cm}	0.06	9.38	-1.08	0.52	1.74	0.49	4.28	-0.54	0.59	1.74
v_{rms}	1.59	7.94	-2.20	1.99	5.97	-0.05	4.38	-2.12	0.18	2.23
v_c	2.69	6.98	-1.14	2.89	6.68	0.35	3.85	-1.48	0.25	2.01
ρ_o	16.2	38.1	-2.22	8.33	26.0	4.03	22.1	-8.82	2.45	13.7
TPM vs. P^3M										
M	1.84	8.65	-3.40	1.75	6.74	0.38	5.14	-1.51	0.39	2.37
v_{cm}	0.22	7.44	-1.19	-0.10	1.20	-0.38	4.26	-0.93	-0.07	0.84
v_{rms}	3.84	7.13	-0.80	3.59	8.94	0.23	3.82	-1.83	-0.03	2.16
v_c	6.20	8.93	1.02	5.11	10.7	0.32	4.06	-1.94	-0.24	2.00
ρ_o	29.4	39.2	3.45	20.0	47.2	13.7	35.0	-2.46	6.82	19.0
TPM ($\Delta t < \sqrt{0.05a^3\epsilon/g}$) vs. Tree										
M	-0.25	8.11	-4.70	-0.08	4.26	-0.63	5.82	-2.73	-0.33	2.18
v_{cm}	-0.01	8.19	-0.97	0.34	1.65	0.76	7.10	-0.68	0.45	1.76
v_{rms}	1.55	7.83	-3.07	1.99	5.36	-1.00	3.86	-2.65	-0.62	1.28
v_c	1.74	6.80	-2.61	1.50	5.69	-0.77	3.85	-2.44	-0.49	1.03
ρ_o	12.7	30.5	-6.24	6.89	26.7	-0.75	24.3	-12.9	-4.16	6.49

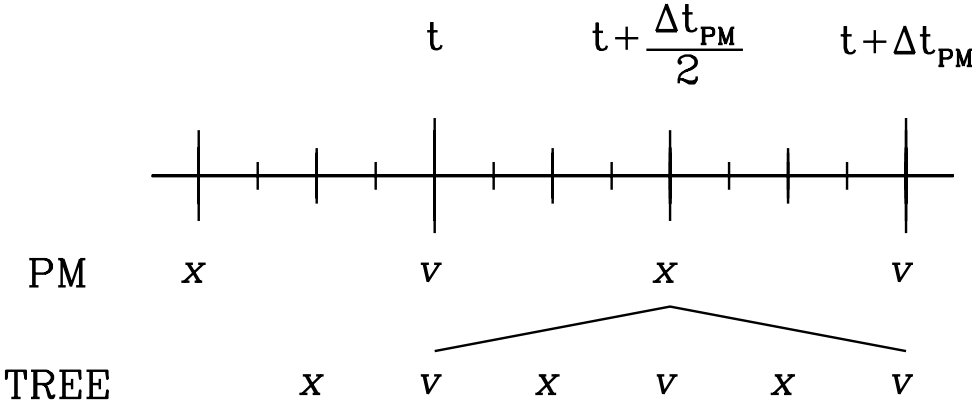

Fig. 1.— Diagram of time stepping: velocities are updated at the times marked v and accelerations at times marked x.

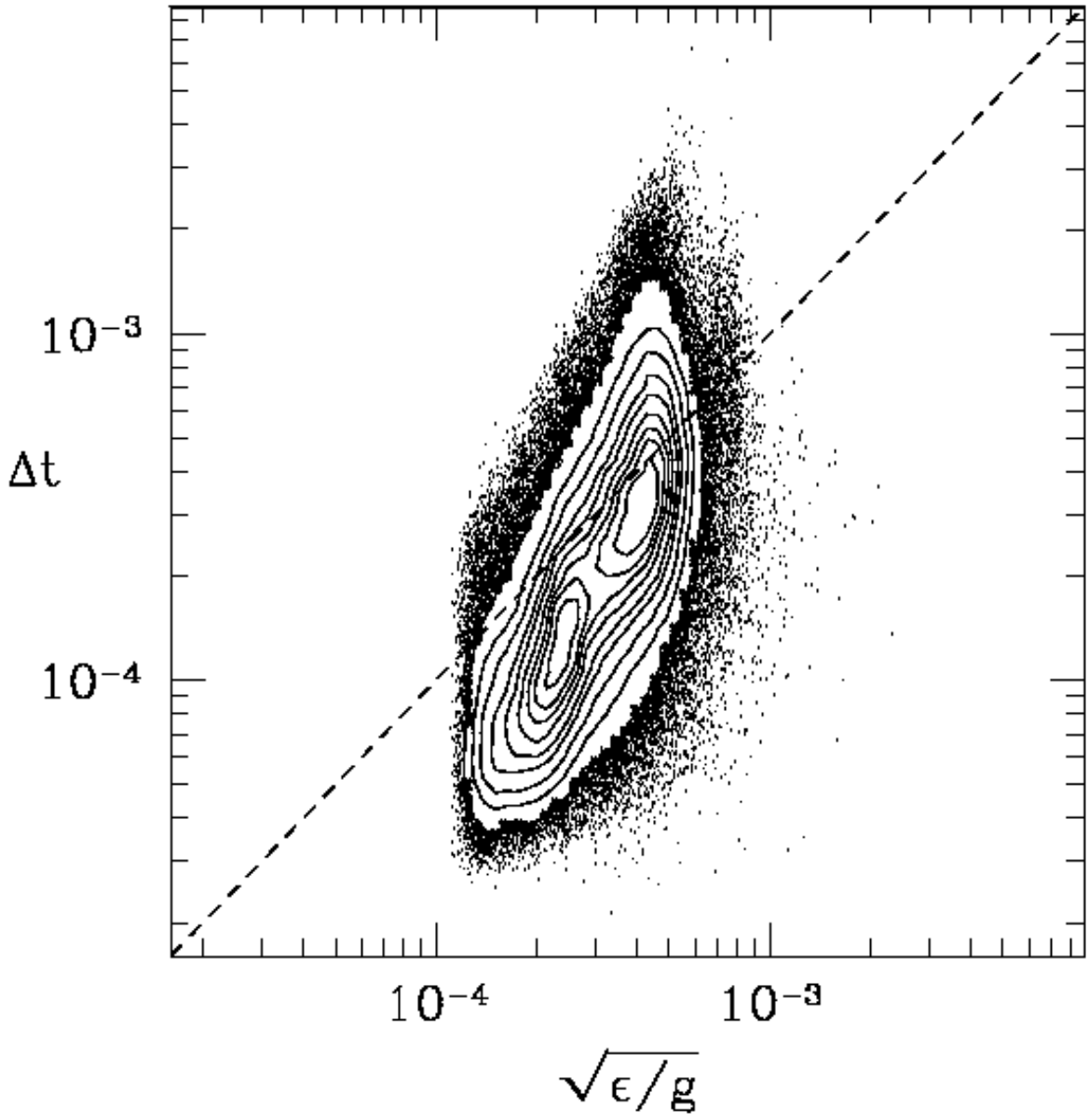

Fig. 2.— Comparison of the time step Δt returned by eq. [6] with that determined by $\eta\sqrt{\epsilon/g}$ for particles in the largest tree of a cosmological simulation at $z = 0$. Each contour level contains one tenth of the particles, and the remaining tenth are plotted as points; the dashed line shows when the two are equal. Code units (in which G, the total mass, and the box length are all unity) are used, with $\eta = 1$.

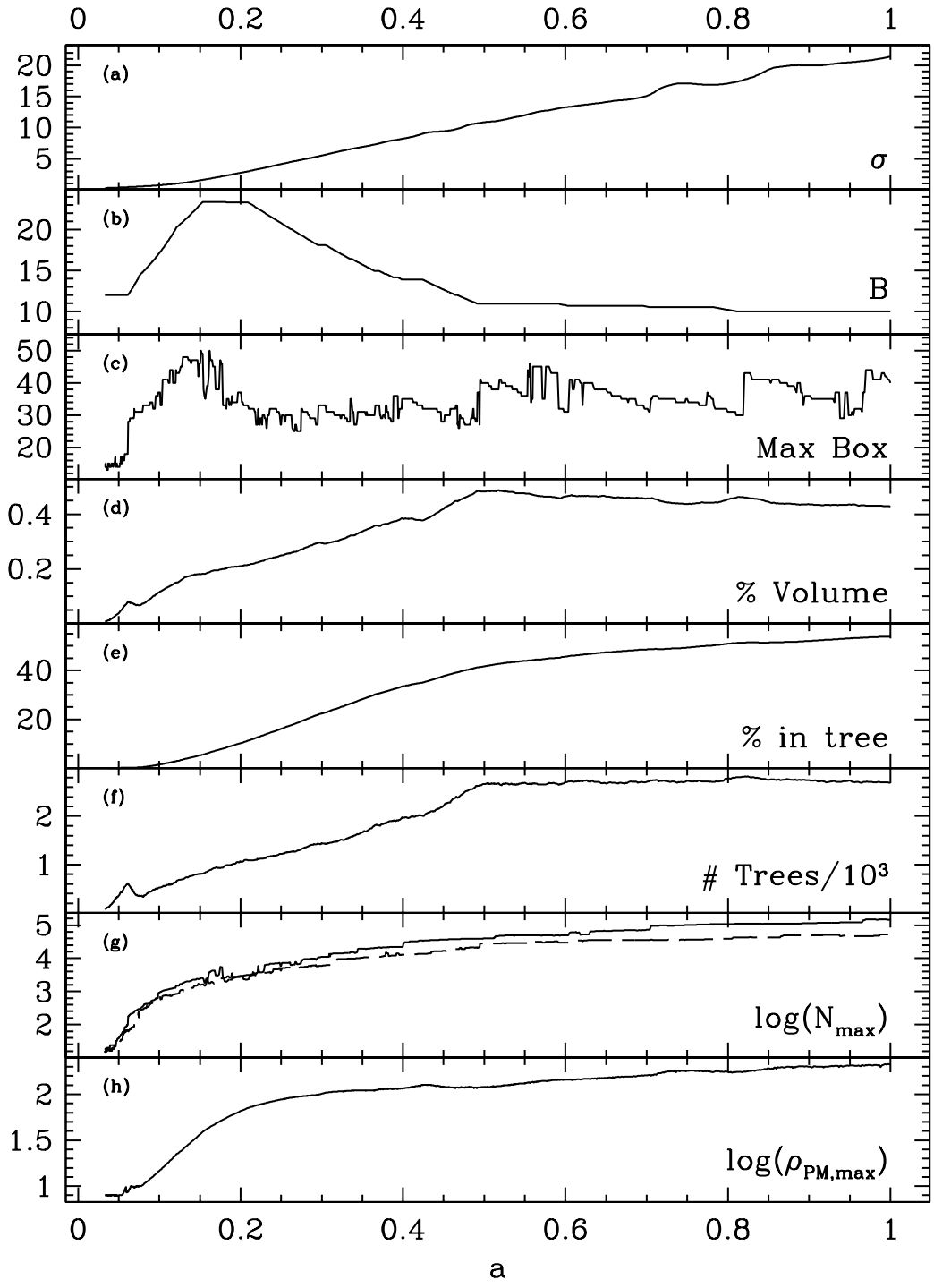

Fig. 3.— Overview of typical run: (*a*) dispersion of PM cell densities (where the mean=1), (*b*) value of B used in eq. [5], (*c*) maximum tree subvolume size, (*d*) % of total volume in trees, (*e*) % of all particles in trees, (*f*) number of trees (in thousands), (*g*) log of number of particles in the first and third most massive trees, (*h*) log of densest PM-only cell.

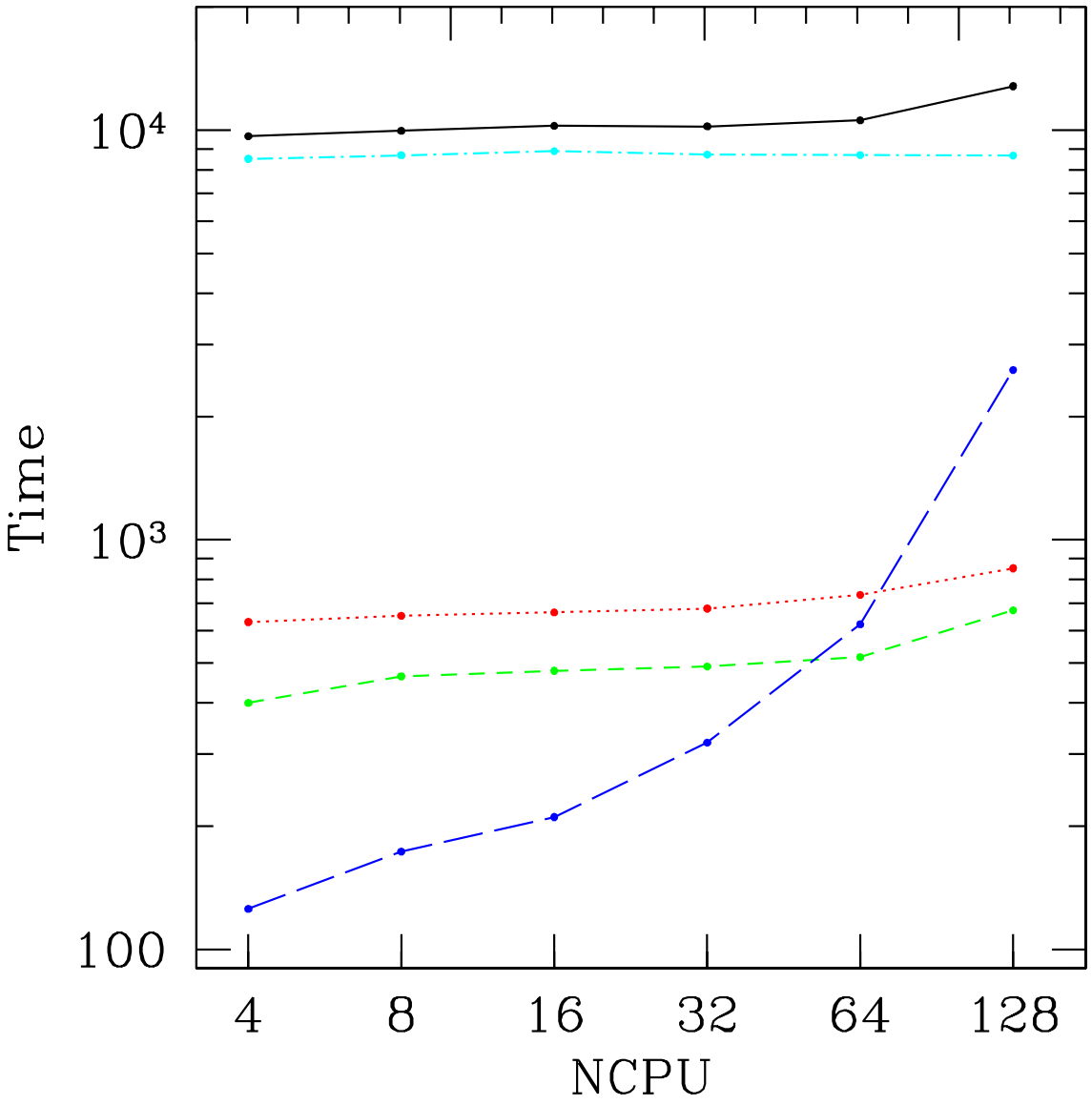

Fig. 4.— Scaling with number of processors, NCPU. The time shown is wallclock time in seconds multiplied by NCPU, so perfect scaling would be a horizontal line. From top to bottom at NCPU=4: total, tree evolution, PM, domain decomposition plus other tree overhead, and time in communication plus load imbalance. The test case is standard LCDM at redshift z=0.16, with N=256^3 particles in a 320h^{-1}Mpc cube.

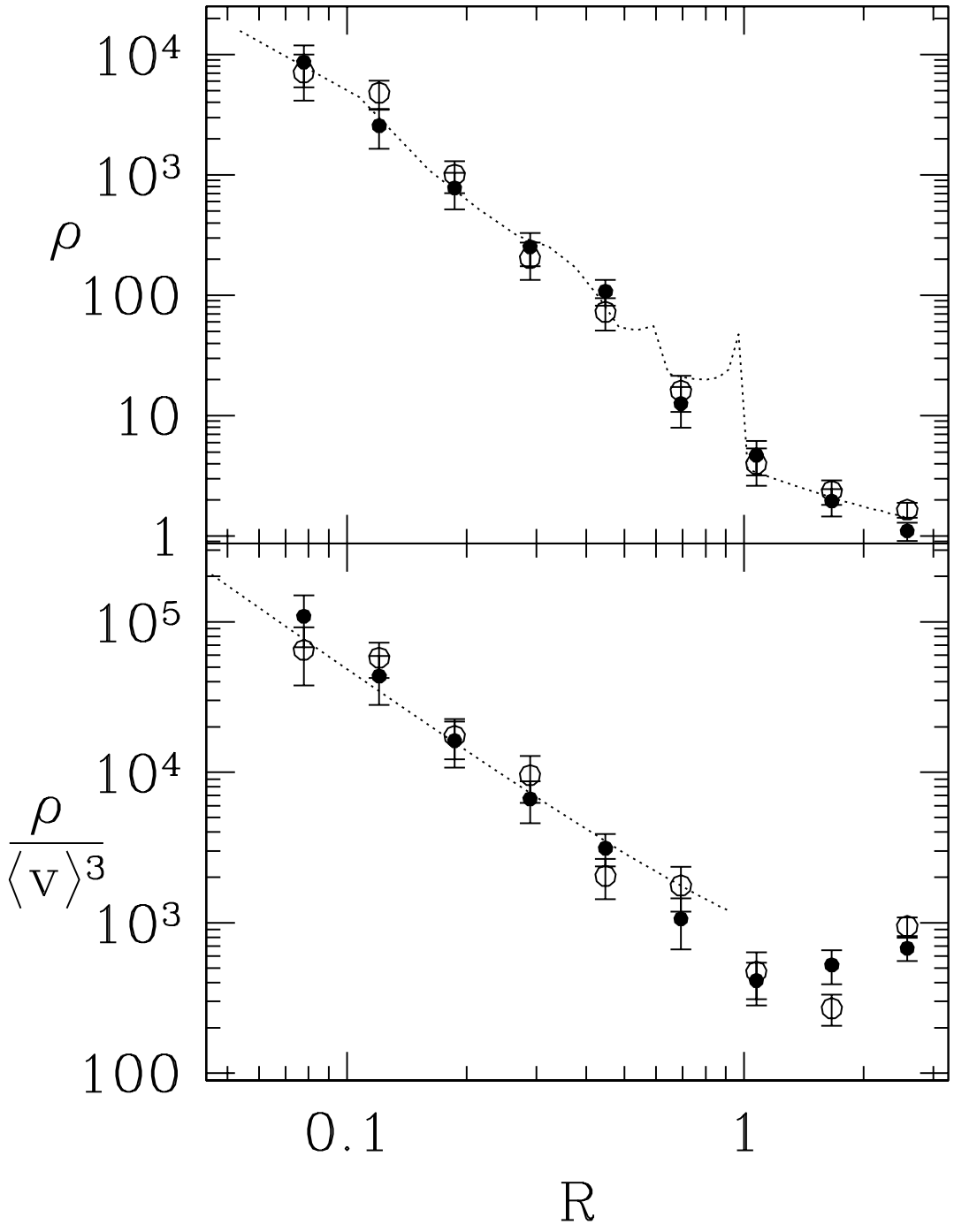

Fig. 5.— The final state of the spherical overdensity infall test. Filled circles: TPM run; open circles: P^3M run; dotted lines: Bertschinger's analytic solution.

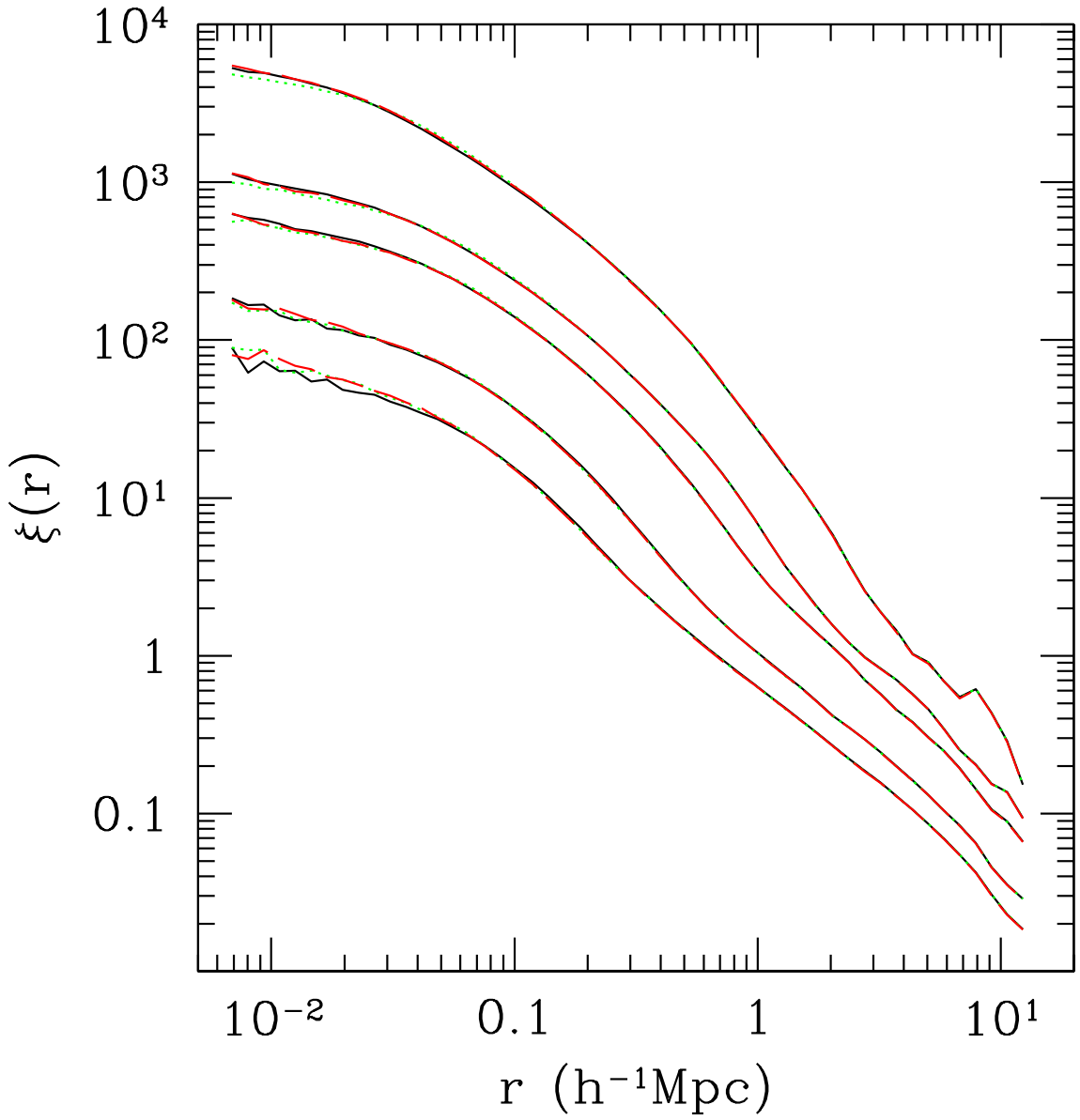

Fig. 6.— Particle–particle correlation function for TPM (solid), P^3M (dotted), and tree (dashed lines) codes. From top to bottom: $z = 0$, 1, 2, 3, and 4.

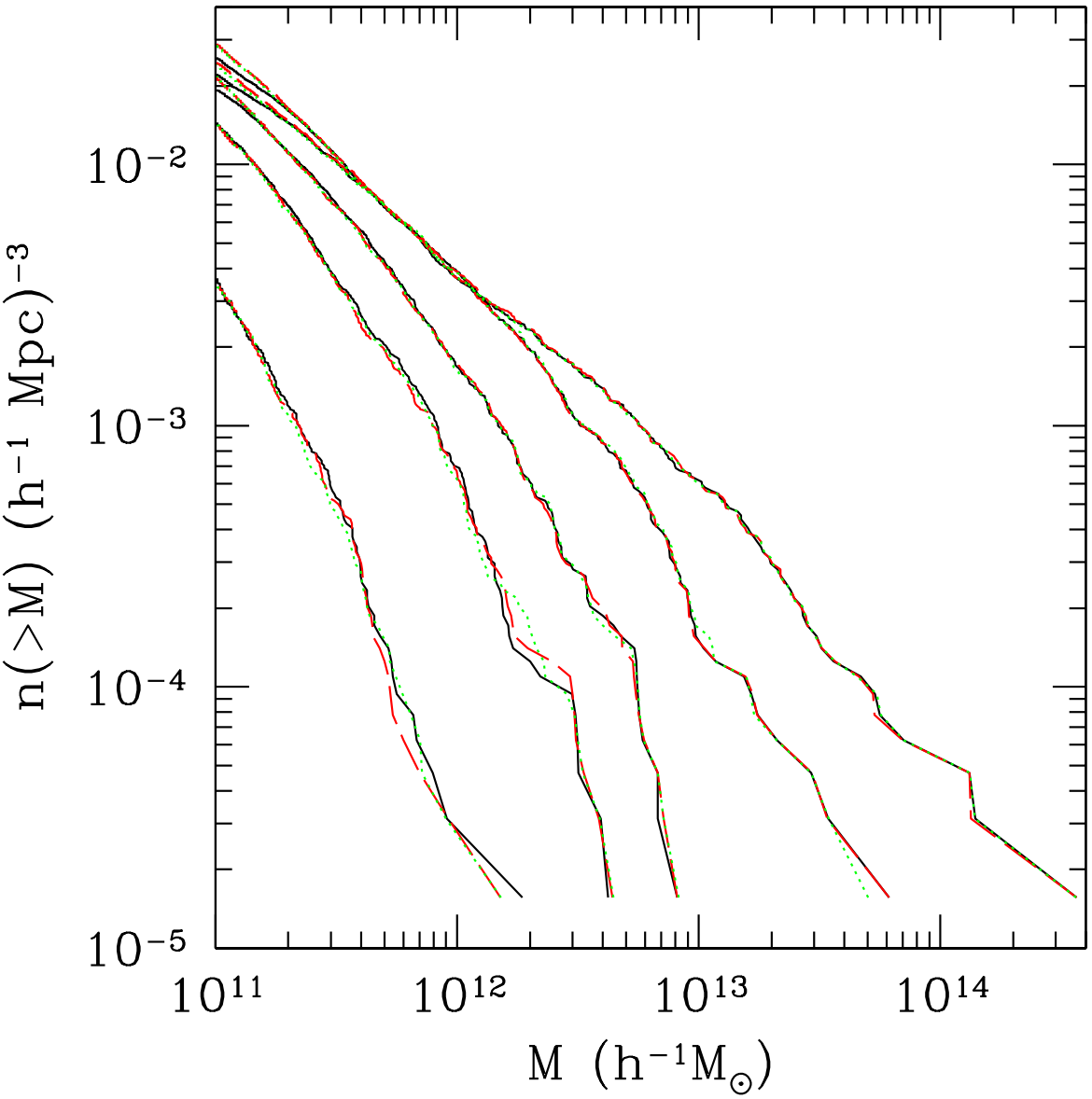

Fig. 7.— Cumulative FOF halo mass function for TPM (solid), P^3M (dotted), and tree (dashed) codes. From top to bottom: $z = 0$, 2, 3, 4, and 6.

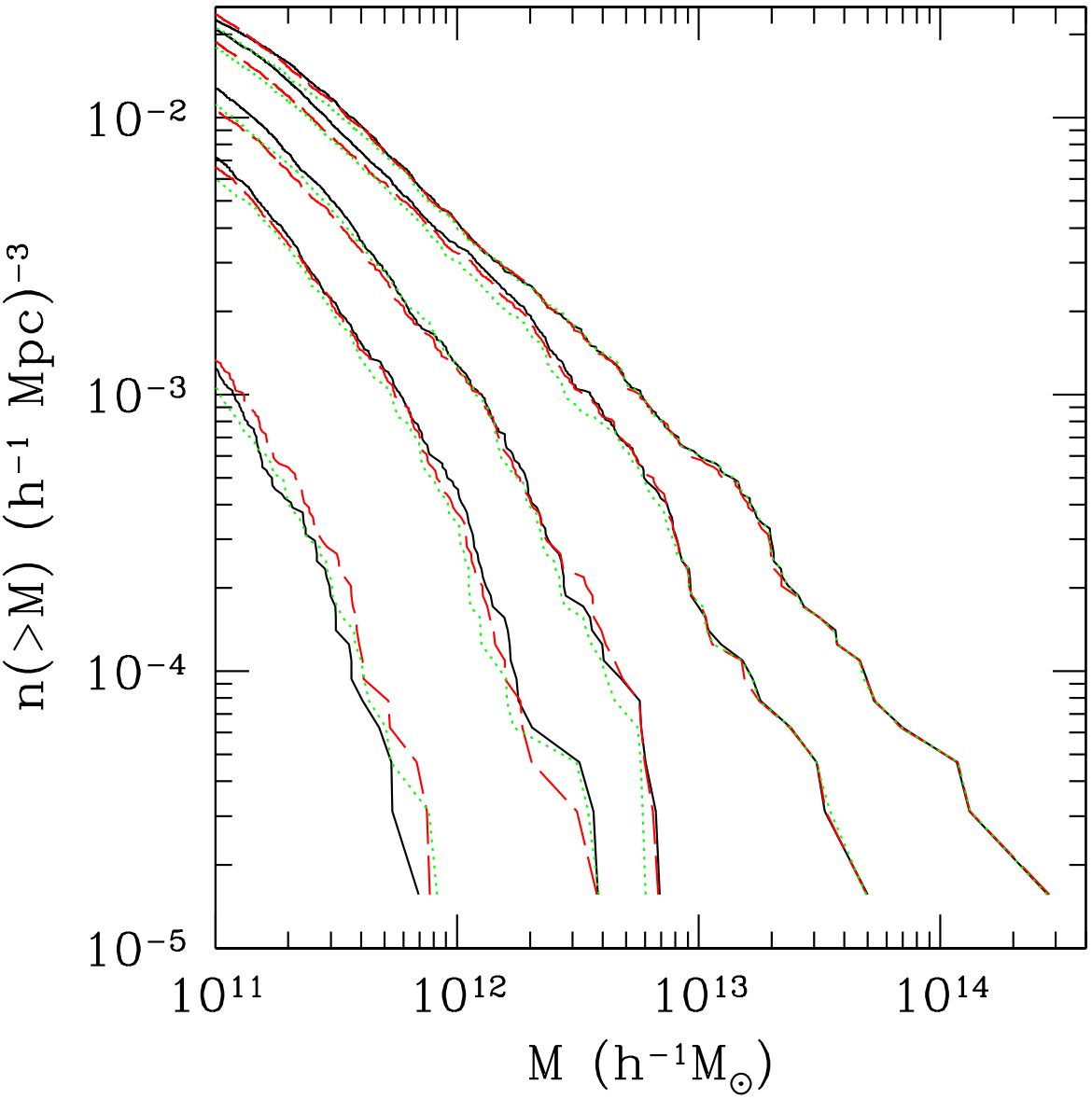

Fig. 8.— Cumulative BDM halo mass function for TPM (solid), P^3M (dotted), and tree (dashed) codes. From top to bottom: $z = 0$, 2, 3, 4, and 6.

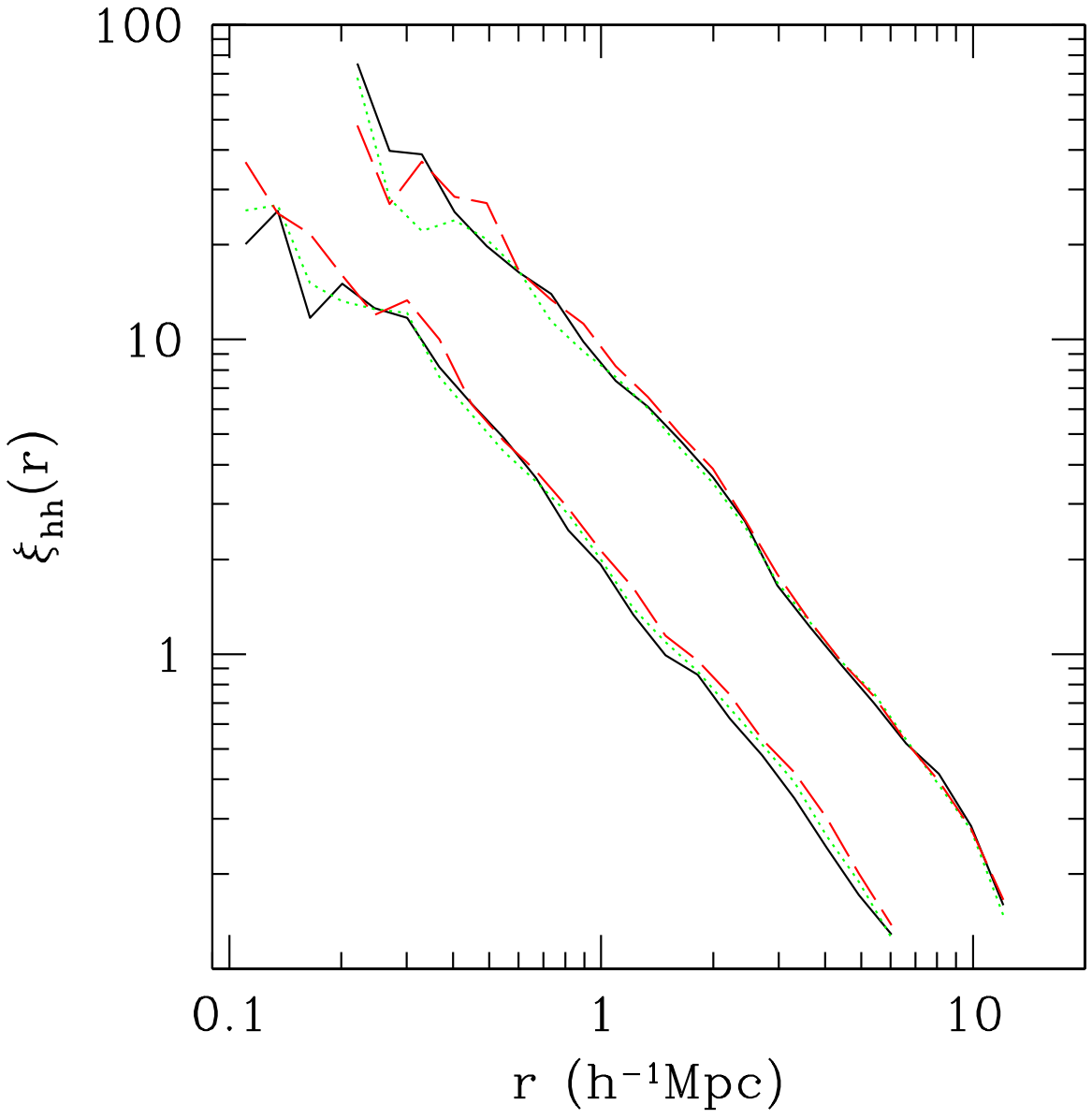

Fig. 9.— The halo–halo correlation function for all halos with 40 particles or more (mass $> 10^{11}h^{-1}M_{\odot}$) at z=0 and 1 (using physical radius); TPM (solid), P^{3}M (dotted), and tree (dashed) codes.

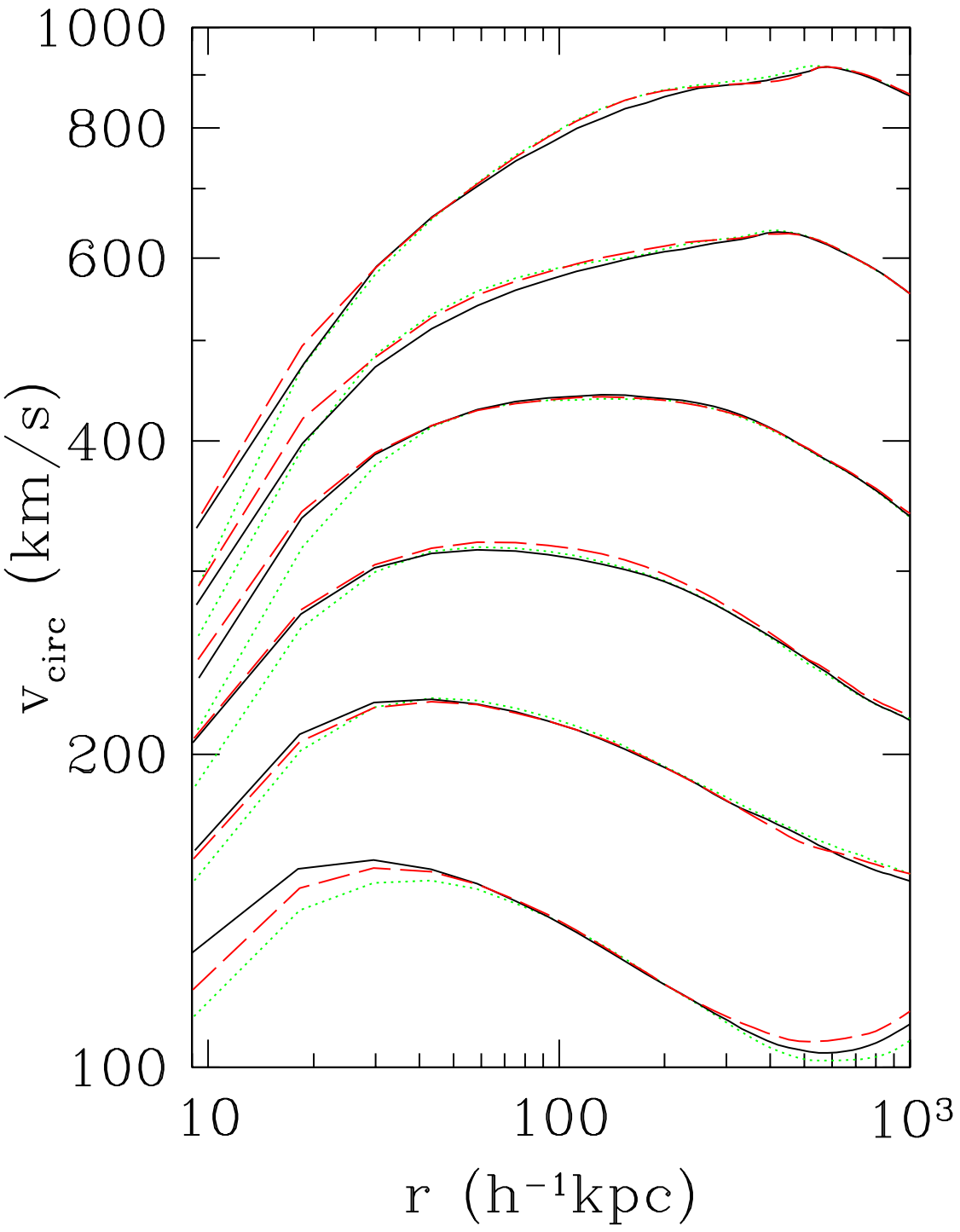

Fig. 10.— Average circular velocity profiles for TPM (solid), P^3M (dotted), and tree (dashed) codes. Each bin spans a factor of $\sqrt{10}$, and the lowest mass used is $3.81 \times 10^{11} h^{-1} M_\odot$ (150 particles).

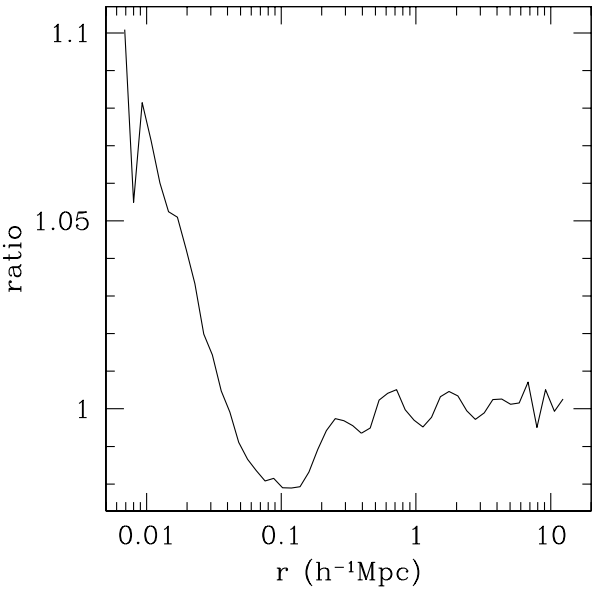

Fig. 11.— Ratio of the particle–particle correlation function $\xi(r)$ for the standard TPM run (using eq. [6]) to that for the run with a less stringent time step criterion.

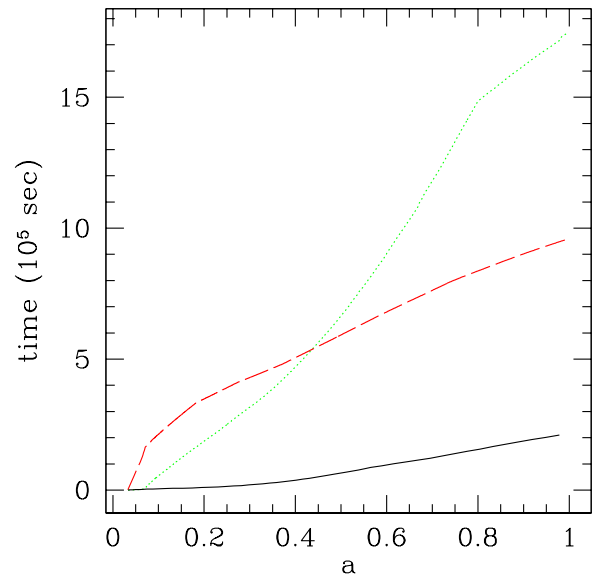

Fig. 12.— Wall-clock time (in units of 10^5 seconds) used by TPM (solid), P^3M (dotted), and tree (dashed) codes as a function of expansion parameter for a standard LCDM model. All codes were run on four processors of an SGI Origin 2000.